**Unit- 3 and 6**

**Unix and Shell Programming Lab (BCAC691)**

Unix Commands

1. who, pwd, mkdir, rmdir, cd, ls, touch, cp, mv, rm
2. chmod, cal, file, sort, grep, man, lpr, passwd, clear, history.
3. cat, date, echo, ftp, head, more, tail, tar, wc, find,
4. sed, awk, vim, gzip, unzip, shutdown, kill, ifconfig, mount, host, wait, sleep.
5. **who: Shows information about users who are currently logged in.**

**Example: who**

1. **pwd: Displays the present working directory.**

**Example: pwd**

1. **mkdir: Creates a new directory.**

**Example: mkdir new\_directory**

1. **rmdir: Removes a directory.**

**Example: rmdir directory\_to\_remove**

1. **cd: Changes the current directory.**

**Example: cd /path/to/directory**

1. **ls: Lists the files and directories in the current directory.**

**Example: ls**

1. **touch: Creates an empty file or updates the timestamp of an existing file.**

**Example: touch new\_file.txt**

1. **cp: Copies files or directories from one location to another.**

**Example: cp file\_to\_copy destination\_directory**

1. **mv: Moves files or directories from one location to another.**

**Example: mv file\_to\_move destination\_directory**

**10.rm: Removes files or directories.**

**Example: rm file\_to\_remove**

**11.chmod: Changes the permissions of a file or directory.**

**Example: chmod 755 file\_name**

**12.cal: Displays a calendar for a specific month or year.**

**Example: cal**

**13.file: Determines the file type.**

**Example: file file\_name**

**14.sort: Sorts the lines of a text file.**

**Example: sort file\_to\_sort**

**15.grep: Searches for a specific pattern in files.**

**Example: grep "pattern" file\_to\_search**

**16.man: Displays the manual for a specific command.**

**Example: man command\_name**

**17.lpr: Sends files to a printer.**

**Example: lpr file\_to\_print**

**18.passwd: Allows a user to change their password.**

**Example: passwd**

**19.clear: Clears the terminal screen.**

**Example: clear**

**20.history: Shows a list of previously executed commands.**

**Example: history**

**21.cat: Displays the content of a file.**

**Example: cat file\_name**

**22.date: Displays the current date and time.**

**Example: date**

**23.echo: Prints text or variables to the terminal.**

**Example: echo "Hello, World!"**

**24.ftp: Transfers files over FTP (File Transfer Protocol).**

**Example: ftp**

**25.head: Displays the first few lines of a file.**

**Example: head file\_name**

**26.more: Displays content page by page in the terminal.**

**Example: more file\_name**

**27.tail: Displays the last few lines of a file.**

**Example: tail file\_name**

**28.tar: Archives files into a tarball (.tar) or extracts files from a tarball.**

**Example: tar -cvf archive.tar files\_to\_archive**

**29.wc: Counts lines, words, and characters in a file.**

**Example: wc file\_name**

**30.find: Searches for files or directories based on specified criteria.**

**Example: find /path/to/search -name "file\_pattern"**

**31.sed: Stream editor for modifying or manipulating text.**

**Example: sed 's/old\_pattern/new\_pattern/' file\_name**

**32.awk: A versatile programming language for text processing.**

**Example: awk '{print $1}' file\_name**

**33.vim: A text editor.**

**Example: vim file\_name**

**34.gzip: Compresses files using the gzip algorithm.**

**Example: gzip file\_name**

**35.unzip: Extracts files from a ZIP archive.**

**Example: unzip archive.zip**

**36.shutdown: Shuts down or restarts the system.**

**Example: shutdown -h now (to shut down immediately)**

**37.kill: Terminates processes by ID or name.**

**38.Example: kill process\_id**

**39.ifconfig: Displays or configures network interface parameters.**

**Example: ifconfig**

**40.mount: Mounts a filesystem.**

**Example: mount /dev/sdb1 /mnt**

**41.host: Performs DNS lookup and displays information about a domain.**

**Example: host domain\_name**

**42.wait: Pauses execution until all background processes finish.**

**Example: wait**

**43.sleep: Delays execution for a specified amount of time (in seconds).**

**Example: sleep 5 (to sleep for 5 sec)**

**Write a shell script to display date and time?**

**#!/bin/bash**

**# Display the current date and time**

**Date**

**Write a shell script to display your name?**

**#!/bin/bas**

**# Display the user's name**

**echo "Your Name"**

**Write a shell script to add two number?**

**#!/bin/bash**

**# Input the two numbers**

**echo "Enter the first number:"**

**read x**

**echo "Enter the second number:"**

**read y**

**# Add the two numbers**

**sum=$((x + y))**

**# Display the result**

**echo "The sum of $x and $y is: $sum"**

**Write a shell script to multiply two number?**

**#!/bin/bash**

**# Input the two numbers**

**echo "Enter the first number:"**

**read x**

**echo "Enter the second number:"**

**read y**

**# Multilication the two numbers**

**sum=$((x \* y))**

**# Display the result**

**echo "The sum of $x and $y is: $sum"**

**Write a shell script to create a for loop?**

**#!/bin/bash**

**# Using a for loop to display numbers from 1 to 5**

**for ((i=1; i<=5; i++))**

**do**

**echo $i**

**done**

**Write a shell script to calculate factorial?**

**#!/bin/bash**

**# Input the number**

**echo "Enter a number:"**

**read num**

**factorial=1**

**# Calculate the factorial**

**for ((i=1; i<=num; i++))**

**do**

**factorial=$((factorial \* i))**

**done**

**# Display the result**

**echo "The factorial of $num is: $factorial"**

**Write a shell script to swap two numbers?**

**#!/bin/bash**

**# Take input for the two numbers**

**echo "Enter the first number:"**

**read num1**

**echo "Enter the second number:"**

**read num2**

**echo "Before swapping:"**

**echo "First number: $num1"**

**echo "Second number: $num2"**

**# Swapping using a temporary variable**

**temp=$num1**

**num1=$num2**

**num2=$temp**

**echo "After swapping:"**

**echo "First number: $num1"**

**echo "Second number: $num2"**

**Write a shell script to create a function?**

**#!/bin/bash**

**# Define the function**

**name() {**

**echo "Hello, $1!" # $1 represents the first argument passed to the function**

**}**

**# Call the function**

**greet\_user "suman"**

**Write a shell script to create an array?**

**#!/bin/bash**

**# Define an array**

**my\_array=("apple" "banana" "orange" "grape" "kiwi")**

**# Print the elements of the array**

**echo "Elements of the array:"**

**for element in "${my\_array[@]}"; do echo "$element"; done**

**Write a shell script to count the length of a string?**

**#!/bin/bash**

**# Prompt the user to enter a string**

**echo "Enter a string:"**

**read input\_string**

**# Calculate the length of the string**

**string\_length=${#input\_string}**

**# Print the length of the string**

**echo "Length of the string is: $string\_length"**

**Write a shell script to concatenate strings?**

**#!/bin/bash**

**# Define two strings**

**string1="Hello"**

**string2="World"**

**# Concatenate the strings**

**concatenated\_string="$string1 $string2"**

**# Print the concatenated string**

**echo "Concatenated string: $concatenated\_string"**

**Write a shell script to read a file?**

**#!/bin/bash**

**# Prompt the user to enter the file name**

**echo "Enter the file name:"**

**read filename**

**# Check if the file exists**

**if [ ! -f "$filename" ]; then**

**echo "File '$filename' does not exist."**

**exit 1**

**fi**

**# Read the file line by line and print its content**

**echo "Content of the file '$filename':"**

**while IFS= read -r line; do**

**echo "$line"**

**done < "$filename"**

**Write a shell script to delete a file?**

**#!/bin/bash**

**# Prompt the user to enter the file name**

**echo "Enter the file name to delete:"**

**read filename**

**# Check if the file exists**

**if [ ! -f "$filename" ]; then**

**echo "File '$filename' does not exist."**

**exit 1**

**fi**

**# Delete the file**

**rm "$filename"**

**echo "File '$filename' has been deleted."**

**Write a shell script to append to a file?**

**#!/bin/bash**

**# Prompt the user to enter the file name**

**echo "Enter the file name to append to:"**

**read filename**

**# Check if the file exists**

**if [ ! -f "$filename" ]; then**

**echo "File '$filename' does not exist. Creating a new file."**

**touch "$filename"**

**fi**

**# Prompt the user to enter the text to append**

**echo "Enter the text to append:"**

**read text\_to\_append**

**# Append the text to the file**

**echo "$text\_to\_append" >> "$filename"**

**echo "Text has been appended to '$filename'."**

**Write a shell script program to print the current process id?**

**#!/bin/bash**

**# Print the current process ID**

**echo "Current process ID: $$"**